Name: *Student id:* *Section:* *Serial#:*

QUESTION #	1	2	3	4	5	TOTAL
MAX POINTS	15	20	8	20	20	
POINTS EARNED						

UNIVERSITY OF BAHRAIN COLLEGE OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE TIME: 90 MINUTES
ITCS242: ASSEMBLY LANGUAGE PROGRAMMING SECOND TEST DATE: DEC 25, 2014
****

QUESTION ONE: Answer each of the three questions as required. **{15 pts}**

1) Give no more than 5 instructions to swap the last pushed two words on the top of the stack. **{4 pts}**

```
pop     ax
pop     bx
push    ax
push    bx
```

2) Give no more than 5 instructions to divide the unsigned value in **SI:BX:DI** by 1024. **{4 pts}**

```
     mov    ecx, 10
l2:  shr    si, 1
     rcr    bx, 1
     rcr    di, 1
     loop   l2
```

```
shrd    di, bx, 10
shrd    bx, si, 10
shr     si, 10
```

3) Write the needed instructions to convert lower-case letters in a string ***strX*** into upper-case.

strX byte "aGb9,56trY#Q... " . **{7 pts}**

```
     mov    ecx, sizeof strx
     mov    ebx, 0
l1:  cmp    strx[ebx], "a"
     jb     l2
     cmp    strx[ebx], "z"
     ja     l2
     sub    strx[ebx], 32
l2:  inc    ebx
     loop   l1
```

Name: *Student id:* *Section:* *Serial#:*

QUESTION TWO: Write a complete Assembly program that implements the following algorithm. **{20 pts}**

1) Begin the program and initialize counters **even**, **odd,** and variables **sumEv, sumOdd** to zero.
2) Input data : prompt the user to read input values from the keyboard
   2.1. display a message that asks the user to enter the next **signed short** number.
   2.2. read the next value into the variable **num**.
3) While (**num** is not negative) do {
   3.1. if **num** is odd do
      increment the counter **odd** and add **num** to **sumOdd**.
   else
      increment the counter **even** and add **num** to **sumEv**.
   3.2. display a message that asks the user to enter the next signed short number.
   3.3. read the next value into the variable **num**.
   3.4. go to step 3.
   }
4) Display the results and finish your program.
   4.1. display the counts of odd and even values each at the beginning at a new line.
   4.2. display the sums of odd and even values each at the beginning at a new line.

```
          INCLUDE   IRVINE32.INC
          .DATA
M1        byte      "Enter the next signed short number please: "
even      sword     0         ; Counters
odd       sword     0
sumOdd    sdword    0         ; Sums
sumEv     sdword    0

          .CODE
MAIN      PROC
          LEA       EDX, M1
          CALL      WRITESTRING
          CALL      READINT
     WH:  CMP       EAX, 0
          JL        DIS
          BT        AX, 0
          JC        L1              ;is the next value odd?
          INC       EVEN
          ADD       SUMEV, EAX
          JMP       L2
     L1:  INC       ODD
          ADD       SUMODD, EAX
     L2:  CALL      WRITESTRING
          CALL      READINT
          JMP       WH
     DIS: CALL      CRLF
          MOVSX     EAX, ODD
          CALL      WRITEINT
          MOV       AL, 9
          CALL      WRITECHAR
          MOVSX     EAX, EVEN
          CALL      WRITEINT
          MOV       AL, 9
          CALL      WRITECHAR
          CALL      CRLF
          MOV       EAX, SUMODD
          CALL      WRITEINT
          MOV       AL, 9
          CALL      WRITECHAR
          MOVSX     EAX, SUMEV
          CALL      CRLF
          EXIT
MAIN      ENDP
          END       MAIN
```

Name: *Student id:* *Section:* *Serial#:*

QUESTION THREE:

What will be in the specified registers after executing each of the following codes? **{ 8 pts}**

a)
```
MOV   AX, 9E7DH
XOR   AX, 55FFH
```

AX = **CB 82** H

b)
```
MOV   BX, 284AH
MOV   AX, 7F9CH
SHLD  BX, AX, 8
```

BX = **4A 7F** H

c)
```
MOV   SP, 3C7AH
PUSH  EBX
PUSH  CX
```

SP = **3C 74** H

d)
```
MOV   AL, 0C0H
MOV   BL, 2AH
IMUL  BL
```

AX = **F5 80** H

e)
```
MOV  AX,3F4AH
MOV  BX,6750H
TEST AX, BX
ROR  AX, 4
```

AX = **A3 F4** H

f)
```
MOV  CX,3F04H
MOV  BX,0C9A7H
SAR  BX, CL
```

BX = **FC 9A** H

g)
```
MOV  AX,4AC7H
MOV  BX,0FFFFH
CWD
IDIV BX
```

AX = **B5 39** H

h)
```
MOV  AX,3FFFH
MOV  BX,7F6AH
CMP  AL, AH
JB   L3
DEC  BL
JMP  L4
L3:DEC  BH
L4:
```

BX = **7F 69** H

QUESTION FOUR: Convert each of the following C++ codes into Assembly language.

1) *signed byte x ,y ;* ***// You have to properly define*** *x, y, and f* **{6 pts}**
*f = (x / y) * (x % y);*

```
x       sbyte       ?
y       sbyte       ?
f       sword       ?

movsx   ax, x
idiv    y               ; x / y
imul    ah              ; ax = (x/y)* (x%y)
mov     f, ax
```

2) **{6 pts}**

```
void funU (short *a, short *b)
{   short t = *a;
    *a = *b;
    *b = t;
}
```

```
funU    proc    uses esi edi ax, a: ptr word, b: ptr word
        mov     esi, a
        mov     edi, b
        mov     ax, [esi]
        xchg    ax, [edi]
        mov     [esi], ax
        ret
funU    endp
```

3) **{8 pts}**

```
int k=0, x[20]= {10, -12,...};
while (k < 20)
{ if (x[k] < 0)
    cout << x[k] << endl;
  k++;
}
```

```
        .data
x       sdword  10, -12, ...
        .code
        mov     ebx, 0              ; index
wh:     cmp     ebx, lengthof x
        jGE     fin
        bt      x[4*ebx], 15
        jnc     L2
        mov     ax, x[4*ebx]
        call    writeint
        call    crlf
L2:     inc     ebx
        jmp     wh
fin:
```

Name: *Student id:* *Section:* *Serial#:*

QUESTION FIVE: **{20 pts}**

Write a complete Assembly program that defines in the data segment two arrays **arrX** and **arrY** each consisting of 80 signed words. The program consists of the two procedures described as follows:

a) The procedure **FUNX** accepts 2 parameters x and y of **short** types, calculates and returns the value of f as shown below. *(Write the procedure FUNX in a form that allows using invoke statement)*

$$f = \begin{cases} x - y & if\ x \geq y \\ 4 * y & if\ x < y \end{cases}$$

b) The procedure **main** fills arrays **arrX** and **arrY** by generating 80 random values for each array in the range from −20 to +20, calls the procedure **FUNX** to calculate the value of f for the corresponding values of **arrX** and **arrY**. Values of f must be stored in array named ***resf*** that you have to define.

```
            INCLUDE    IRVINE32.INC
            .DATA
arrX        sword      80 DUP(?)
arrY        sword      80 DUP(?)
resF        sword      80 DUP(?)
            .CODE
;     ********************************************
FUNX        PROC       USES EBX EAX, X:WORD, Y: WORD, F: PTR WORD
            MOV        ESI, F            ;    MOV       ESI, F
            MOV        AX, X             ;    MOV       AX, X
            CMP        AX, Y             ;    MOV       BX, Y
            JGE        L2                ;    SUB       AX, Y
            MOV        BX, Y             ;    MOV       [ESI], AX
            SAL        BX, 2             ;    CMP       X, BX
            MOV        [ESI], BX         ;    JGE       L3
            JMP        L3                ;    SAL       BX, 2
      L2:   SUB        AX, Y             ;    MOV       [ESI], BX
            MOV        [ESI], AX         ; L3:RET
      L3:   RET
FUNX        ENDP
;     ********************************************
MAIN        PROC
            CALL       RANDOMIZE
            MOV        ECX, LENGTHOF ARRX                    EXIT
            MOV        EBX, 0                             MAIN ENDP
LU:         MOV        EAX, 41                               END       MAIN
            CALL       RANDOMRANGE
            SUB        EAX, 20
            MOV        ARRX[2*EBX], AX
            MOV        EAX, 41
            CALL       RANDOMRANGE
            SUB        EAX, 20
            MOV        ARRY[2*EBX], AX
            INC        EBX
            LOOP       LU
            MOV        ECX, LENGTHOF ARRX
            MOV        EBX, 0
LX:         INVOKE     FUNX, arrX[2*ebx], ARRY[2*EBX], ADDR resF[2*EBX]
            INC        EBX
            LOOP       LX
```